VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Použití deskriptoru ve zpracování obrazu

Usage of Descriptors in Image Processing

2010

Martin Štěpán

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Štěpán**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Použití deskriptoru ve zpracování obrazu
Usage of Descriptors in Image Processing

Zásady pro vypracování:

V současné době existuje mnoho způsobů jak v obraze detekovat objekty. Jedním ze způsobů je použití deskriptorů. Cílem práce je implementace vybraného druhu deskriptoru pro detekci objektů v obraze. Ve své práci proveďte následující:

1. Popište techniku deskriptorů ve zpracování obrazu,
2. po konzultaci s vedoucím práce vyberte jeden deskriptor, který budete implementovat,
3. vybraný deskriptor náležitě popište, implementujte a otestujte,
4. svou práci náležitě zhodnoťte.

Seznam doporučené odborné literatury:

Dle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura**

Datum zadání: 20.11.2009
Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 5. května 2010 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2010 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem této práce je implementace využití deskriptoru ve zpracování obrazu, s využitím knihovny OpenCV. Práce se nejprve zabývá teoretickým popisem několika deskriptorů. Další část je již věnována konkrétnímu deskriptoru - Histogram of Oriented Gradients. Jsou zde popsány jednotlivé kroky algoritmu, včetně klasifikace obrázku. Poslední část se věnuje samotné implementaci deskriptoru Histogram of Oriented Gradients. Postupně je zde popsána implementace jednotlivých kroků deskriptoru. Program je implementován v jazyce C, s využitím knihovny OpenCV.

Klíčová slova: deskriptor, histogram, gradient, Histogram of Oriented Gradients

Abstract

The purpose of this work is to implement an image descriptor, using OpenCV library. The first part of the work is dedicated to the theoretical description of some descriptors. The following part is dedicated to one concrete descriptor – Histogram of Oriented Gradients. Every single part of the algorithm is described, including classification of the image. The last topic is the implementation of the Histogram of Oriented Gradients descriptor itself. The implementation details of each step of the desciptor are described. The program is implemented in C language using OpenCV library.

Keywords: descriptor, histogram, gradient, Histogram of Oriented Gradients

Seznam použitých zkratek a symbolů

C-HOG	–	Circular HOG
DOG	–	Difference of Gaussians
HOG	–	Histogram of Oriented Gradients
LOG	–	Laplacian of Gaussian
R-HOG	–	Rectangular HOG
SIFT	–	Scale-invariant feature transform
SVM	–	Support vector machine

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Digitální zpracování obrazu má v dnešní době, kdy se s počítači setkáme doslova na každém kroku, velký význam. Jedná se o použití počítačových algoritmů na digitální obraz. Digitální zpracování má spoustu výhod oproti analogovému: umožňuje nám aplikovat metody, které nelze aplikovat u analogového zpracování, což nám umožňuje aplikovat větší rozsah algoritmů a také mnohem důmyslnější algoritmy. Příkladem digitálního zpracování obrazu může být například segmentace obrazu nebo odstranění šumu. Digitální zpracování má široké využití. Může být použito například k rozpoznávání vzorů, detekce hran, detekce osob v obraze nebo ve videu, detekce různých jiných objektů,

Detekce lidí na fotografiích představuje náročný problém. Lidé se od sebe můžou lišit v celé řadě věcí. Od změn v oblékání až po samotný vzhled. Také musíme brát v potaz, že lidé mohou být zachyceni v různých pozicích a v různém měřítku. Samotný vzhled člověka není to jediné, co má tedy vliv na výkon samotné detekce. Velký vliv na správnou detekci má také pozadí fotografie - například míra osvětlení.

Automatická detekce osob má širokou škálu využití. Například v robotice, indexování obrazu a videa nebo v oblasti automobilové bezpečnosti.

V této práci je pomocí Histogram of Oriented Gradients deskriptoru představena kompletní detekce lidí na fotografiích. Systém detekce je založen na HOG deskriptoru v kombinaci se Support Vector Machine, určeným pro klasifikaci dat.

1.1 Struktura práce

- Kapitola 2 se zabývá vysvětlením pojmu deskriptor. Je zde popsáno k čemu slouží a na jaké skupiny se dělí. Součástí této kapitoly je také popis několika používaných deskriptorů - Laplacian of Gaussian, Difference of Gaussians, SIFT a Shape context.

- Kapitola 3 se věnuje představení pojmů histogram a gradient. Popisuje o co se jedná a jejich využití.

- Kapitola 4 představuje vybraný deskriptor - Histogram of Oriented Gradients. Nejprve je popsáno několik verzí deskriptoru a následně jsou krok po kroku popsány jednotlivé operace používány při aplikaci tohoto deskriptoru. Použití vybraného deskriptoru je v této práci demonstrováno na detekci lidí. Část této kapitoly je věnována vysvětlení klasifikování dat pomocí algoritmu Support Vector Machine, který byl použit při implementaci pro klasifikaci. Detailně je zde popsán postup, kterým algoritmus zjišťuje zda zadaný obrázek obsahuje zadaný objekt, který jsme chtěli detekovat.

- Kapitola 5 je zaměřena na popis vlastní implementace vybraného deskriptoru. Na úvod je uvedena zmínka o knihovně OpenCV, která je využívána v implementaci. Poté je krok po kroku popsán deskriptor z hlediska samotné implementace. Program je implementován v jazyce C.

- Kapitola 6 je věnována samotnému testování aplikace. Popisuje průběh testování od počáteční detekce válečků, až po samotnou detekci osob na fotografiích, ukázku testovaných dat a zhodnocení výsledků samotného testování.
- Kapitola 7 shrnuje výsledky této práce.

2 Deskriptory

V počitačové terminologii rozumíme pod pojmem deskriptor popis vizuálních vlastností obrázků nebo videa, případně se může jednat o algoritmy nebo aplikace, které nám produkují tyto popisy. Deskriptory popisují základní charakteristiky, jako je tvar, barva nebo textura. Díky těmto popisům můžeme rychle a efektivně vyhledávat vizuální obsah. Vizuální deskriptory jsou rozděleny do dvou hlavních skupin:

- Deskriptory popisující obecné informace: obsahují nízkoúrovňové deskriptory, které nám poskytují informace o tvaru, barvě nebo textuře.
- Deskriptory popisující specifické informace: tyto deskriptory obsahují informace o objektech ve scéně. Konkrétním příkladem může být například rozpoznávání tváří.

V počítačové oblasti se jako blob detektory označují vizuální moduly, jejichž úkolem je detekovat body, případně regiony v zadaném obraze, které jsou buď svetlejší nebo tmavší než okolí. Tyto detektory poskytují doplňující popis struktury obrazu. Jedním z hlavních využití těchto detektorů je poskytnutí bližších informací o regionech, které nebyly získány detektorem hran. Tyto regiony mohou být oblástí zájmu pro další zpracování, může se jednat například o rozpoznání objektů, analýzu histogramu nebo detekci vrcholů s následnou segmentací. Dalším využitím těchto deskriptorů je analýza a rozpoznávání textur.

2.1 Laplacian of Gaussian

Laplacián je metoda založena na druhé prostorové derivaci obrázku. Zvýrazňuje rychlé změny intenzity a proto je často tato metoda využívána pro detekci hran. Často se jako předzpracování obrazu používá Gaussův vyhlazovací filtr. Jako vstup tento operátor bežně používá obrázek převeden do stupně šedi a výstupem bývá zase obrázek ve stupni šedi. Laplacián $L(x, y)$ obrázku s pixely intenzity $I(x, y)$ se vypočítá jako (1):

$$L(x, y) = \frac{\delta^2 I}{\delta x^2} + \frac{\delta^2 I}{\delta y^2} \tag{1}$$

Toto lze vypočítat pomocí konvulenčního filtru.

Vzhledem k tomu, že vstupní obraz je reprezentován jako soubor diskrétních bodů, musíme najít diskrétní konvulenci jádra, které může aproximovat druhé derivace v definici Laplaciánu. Na obrázku 1 jsou zobrazeny dvě bežně používáná malá jádra. Použitím jednoho z těchto jader, je možné vypočítat Laplacián konvulence pomocí standartních metod. Protože jsou tyto jádra velice citlivá na šum, tak se před použitím Laplaceova filtru provádí vyhlazení pomocí Gaussova filtru. Toto předzpracování snižuje úroveň šumu.

Jelikož jsou konvulenční operace asociativní, tak můžem spojit Gaussovo vyhlazování a Laplaciánův filtr do jednoho filtru. Tento hybridní filtr pak aplikujeme na obrázek a dosáhnem stejného požadovaného výsledku. Spojení těchto dvou filtrů má dvě výhody:

0	-1	0
-1	4	-1
0	-1	0

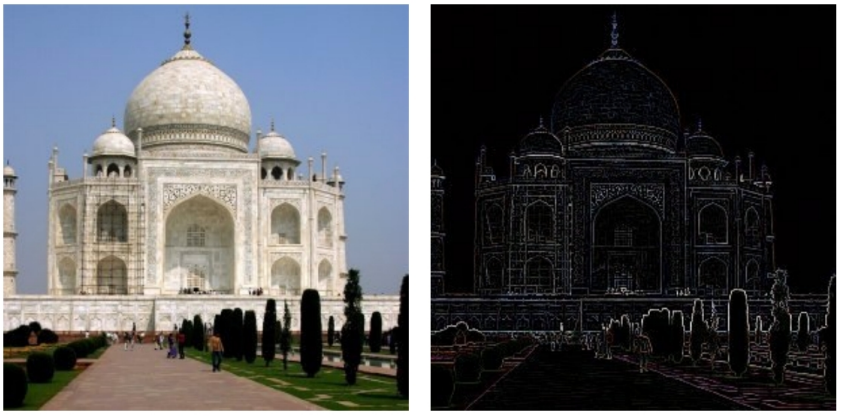
-1	-1	-1
-1	8	-1
-1	-1	-1

Obrázek 1: Dvě bežně používaná malá jádra [33]

- Jelikož Gaussova a Laplacian jádra jsou obvykle mnohem menší, než obraz, tak tato metoda obvykle vyžaduje mnohem méně aritmetických operací.
- LOG jádro může být vypočteno s předstihem a v reálném čase se pak na obrázku počítá jen jedna konvulence.

2-D funkce LOG funkce, se středem v nule a Gaussovou směrodatná odchylkou σ má tvar (2):

$$LOG(x,y) = -\frac{1}{\pi\sigma^4}[1 - \frac{x^2+y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$



Obrázek 2: Použití LOG filtru [30]

2.2 Difference of Gaussians

V počítačové oblasti se jako Difference of Gaussians označuje obrázek ve stupni šedi, na který je aplikován algoritmus, který zahrnuje odečítání jedné rozmazané verze originálního obrázku ve stupních šedi z jiného, méně rozmazané oproti originálnímu. Rozmazané snímky jsou získány konvulencí původního obrázku ve stupních šedi Gaussovými jádry s různými standardními odchylkami. Tento algoritmus se používá pro vylepšení obrazu, například se může jednat o zvýšení viditelnosti hran a dalších podrob-

ností v digitálním obraze. Algoritmus je podobný LOG, obraz je nejprve vyhlazen konvulencí s Gaussovým jádrem s šírkou σ_1 (3):

$$G\sigma_1(x,y) = \frac{1}{\sqrt{2\pi\sigma_1^2}} exp[-\frac{x^2+y^2}{2\sigma^2}] \tag{3}$$

dostanem (4):

$$g_1(x,y) = G\sigma_1(x,y) * f(x,y) \tag{4}$$

S jinou šířkou σ_2, dostanem druhý vyhlazený obraz jako (5):

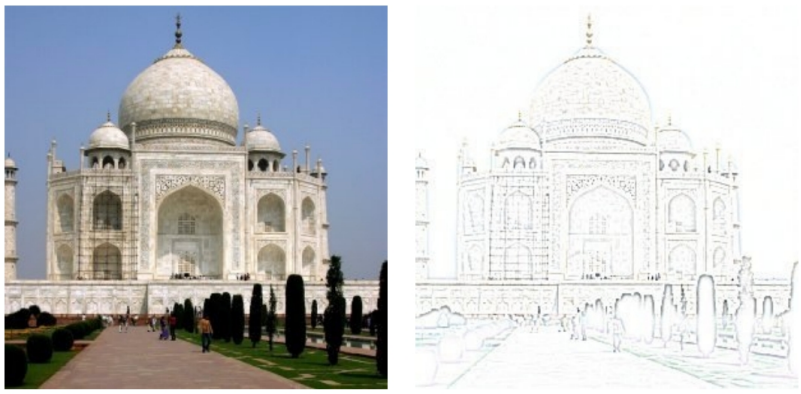
$$g_2(x,y) = G\sigma_2(x,y) * f(x,y) \tag{5}$$

Lze ukázat, že rozdíl těchto dvou Gaussově vyhlazených obrázků, nazývaný Difference of Gaussians, může být použit pro detekci hran v obraze. Tento rozdíl se vypočte jako (6):

$$g_1(x,y) - g_2(x,y) = G\sigma_1 * f(x,y) - G\sigma_2 * f(x,y) = (G\sigma_1 - G\sigma_2) * f(x,y) = DOG * f(x,y) \tag{6}$$

DOG operátor je pak definován jako (7):

$$DOG = G\sigma_1 - G\sigma_2 = \frac{1}{\sqrt{2\pi}}[\frac{1}{\sigma_1} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2} e^{-(x^2+y^2)/2\sigma_2^2}] \tag{7}$$



Obrázek 3: Použití DOG filtru [31]

2.3 Scale-invariant feature transform (SIFT)

Jedná se o algoritmus určený pro detekci a popis lokálních rysů v obraze. Používá se například pro rozpoznávání objektů, rozpoznávání gest nebo sledování videa.

Obrázek 4: SIFT deskriptor - ukázka detekce objektů [11]

Pro každý objekt v obraze lde získat tzv. "popis rysu". Tento popis lze pak použít k porovnání objektů, při vyhledávání objektu v testovacím obrázku. Je velice důležité, aby byly údaje získané z obrazu odolné vůči změnám měřítku, vlivu osvětlení a šumu, aby se dosáhlo spolehlivého rozpoznání.

SIFT deskriptor má čtyři hlavní fáze:

- Scale-Space detekce extrémů
- Lokalizace klíčových bodů
- Přiřazení orientace
- Deskriptor klíčových bodů

V první fázi se používá Gaussův operátor k identifikaci klíčových hodnot. Pro zlepšení výkonnosti při výpočtu se používá Difference of Gaussian operátor. Dostanem rovnici (8):

$$D\left(x,y,\sigma\right)=\left(G\left(x,y,k\sigma\right)-G\left(x,y,\sigma\right)\right)*I\left(x,y\right)=L\left(x,y,k\sigma\right)-L\left(x,y,\sigma\right) \tag{8}$$

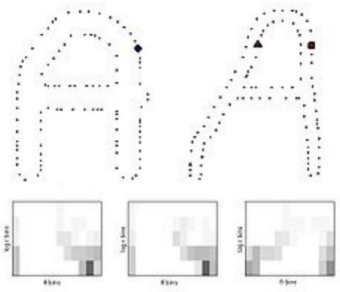
kde $L(x,y,k\sigma)$ je konvulence originálního obrazu $I(x,y)$ Gaussovým rozostřením $G(x,y,k\sigma)$ s měřítkem $k\sigma$.

V druhé fázi vyřadíme klíčové body s nízkým kontrastem a body, u kterých se špatně určuje umístění. Pro výpočet bodů, které budou vyřazeny se používá Hessova matice, která vypočte principiální zakřivení a vyřadí všechny body, které mají větší poloměr než je zakřivení.

V třetím kroku je každému klíčovému bodu přiřazen jeden nebo více směrů, na základě místních směrů gradientu obrazu. Tento krok je klíčový pro dosáhnutí nezávisloti k rotaci, tedy aby deskriptor mohl být reprezentován relativně k orientaci a abychom dosáhli nezávislosti na pootočení obrazu.

2.4 Shape context

Jak již z názvu vyplívá, je tento deskriptor založen na popisu tvarů. Na základě popisu hledaného tvaru se porovnáná podobnost s tvary v obraze - pro každý bod p_i zadaného



Obrázek 5: Shape context - na tomto obrázku lze vidět, že tvar kontextu pro kruh (levý obrázek) a diamant (pravý obrázek) jsou skoro stejné. Oproti tomu, kontextový tvar trojúhelníku (pravý obrázek) je úplně odlišný. [29]

tvaru, chceme najít nejpodobnější bod q_i v jiném tvaru. Nalezení podobného tvaru je jednodušší při použití hustých místních deskriptorů, které snižují riziko dvojznačnosti při porovnávání. Dalším využitím tohoto deskriptoru je rozpoznávání objektů.

Základní myšlenkou je vzít všech n bodů, tvořících tvar objektu a ke každému bodu vytvořit $n - 1$ vektorů, získaných připojením bodu ke všem ostatním bodům. Tyto vektory nám pak udávají popis tvaru v daném bodě. Tím vytvoříme histogram (používá se log-polární histogram) bodu vzhledem ke zbývajícím $n - 1$ bodům (9):

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\} \tag{9}$$

tímto máme definován kontext tvaru bodu p_i.

Aby byl deskriptor použitelný, tak musí být nezávislý vůči změně pozice, změně měřítka a závislý na rotaci obrazu. Nezávislost na změně pozice je dodržena, díky vlastnostem tohoto deskriptoru. Nezávislosti na změně měřítka je dosaženo díky normalizaci všech radiálních vzdáleností, pomocí průměrné vzdálenosti α mezi všemi páry bodů ve tvaru.

Jednotlivé kroky deskriptoru:

- Náhodně se vybere množina bodů, tvořící tvar známého objektu a množina bodů, tvořící tvar neznámého objektu.
- Pro každý bod z vybrané množiny se vypočítá popis tvaru.
- Porovnání každého bodu z množiny bodů známého tvaru s každým bodem množiny neznámého tvaru.
- Vypočet "vzdálenosti tvarů"mezi každým párem bodů. Vzdálenost je určena jako počet transformací, které je potřeba provést, aby tvary byly stejné.
- K identifikaci neznámého tvaru, se porovnává vzdálenost tvaru se vzdálenostmi známých tvarů.

3 Pojmy histogram a gradient

3.1 Jasový histogram

Histogram je graf reprezentující rozložení jasu v obraze. Pro každý jas od černé vlevo, až do bílé vpravo vyjadřuje poměrné zastoupení počtu pixelů. Na vodorovnou osu se vynáší jas a na svislou počet pixelů, které tomuto jasu odpovídají. Počet buněk odpovídá ploše snímku. Histogram nám tedy dává odpověd na tyto otázky:

- Jak velká část obrazu je černá
- Jak velká část obrazu je tmavě šedá
- Jak velká část obrazu je světle šedá
- Jak velká část obrazu je zcela bílá
- atd.

3.2 RGB Histogram

Barevný histogram je reprezentací rozložení barev v obrazu. Vyjadřuje poměrné zastoupení počtu pixelů každého z daných barevných rozsahů v barevném modelu. Není problém udělat histogram každé složky R, G i B samostatně. Histogram složky R potom ukazuje rozložení červené v obraze (od černé až po nejčervenější jakou lze zobrazit), histogram složky G ukazuje rozložení zelené v obraze a histogram složky B pak ukazuje rozložení modré barvy v obraze. Z barevného RGB obrazu lze snadno udělat černobílý obraz. Jde jen o to převést všechny barvy na stupně šedi. Nabízí se jednoduchá metoda, výpočtu absolutního jasu (neboli nových hodnot R, G i B) (10):

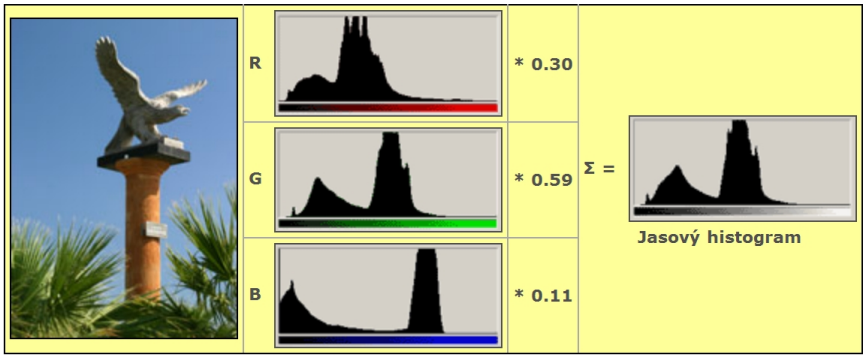
$$Jas = (R + B + G)/3 = 1/3R + 1/3G + 1/3B \tag{10}$$

Lidské oko není stejně citlivé na všechny barvy, na modrou je mnohem méně citlivé než např. na zelenou nebo červenou. Souvisí to s barvou světla z našeho slunce. Proto absolutní jas nemá příliš smysl (oko to prostě vidí jinak) a skutečný jas je definován jako (11):

$$Jas = 0.3R + 0.59G + 0.11B \tag{11}$$

3.3 Gradient

Z obecného pohledu se jedná o směr růstu. Z matematického pohledu se jedná o diferenciální operátor. Výsledkem je vektorové pole vyjadřující směr a velikost největší změny skalárního pole. Z pohledu souřadnicového vyjádření, je v daném místě gradientem vektor. Složky vektoru tvoří jednotlivé parciální derivace funkce vyjadřující dané skalární pole.



Obrázek 6: Příklad RGB histogramu a odpovídajícího jasového histogramu [32]

Parciální derivace představuje derivaci funkce, při které se derivuje pouze vzhledem k jedné z proměnných. Ostatní proměnné jsou považivány za konstanty.

Z matematického hlediska je skalární pole funkce přiřazující skalár (veličinu, která je určená jediným číselným údajem) každému bodu prostoru. Příkladem může být například teplota.

Pro dvourozměrný prostor je gradient (12):

$$grad\,\Theta = (\frac{\delta\Theta}{\delta x}, \frac{\delta\Theta}{\delta y}) \tag{12}$$

Zobecnění pro n-rozměrný prostor lze s pomocí Einsteinova sumarizačního pravidla vyjádřit ve tvaru (13):

$$f = e_i \frac{\delta f}{\delta x_i} \tag{13}$$

kde $x_1, x_2, ..., x_n$ jsou souřadnice a e_i jsou bázové vektory.

4 Histogram of Oriented Gradients

Jedná se o deskriptor používaný v počítačové oblasti a digitálního zpracování obrazu pro detekci objektů. Základní myšlenkou tohoto deskriptoru je, že lokální vzhled objektu a tvar v obrázku lze popsat intenzitou gradientu. Tato metoda je založena na hodnocení sítě normalizovaných místních histogramů. Detektor pracuje v posuvném okně, toto okno je rozděleno do překrývajících se bloků a každý blok je rozdělen do nepřekrývajících se mřížek, ve kterých se počítají jednotlivé histogramy. Toto rozdělení lze vidět na obrázku 7.

Tento deskriptor má několik výhod oproti jiným deskriptorům. Vzhledem k tomu, že výpočet histogramu probíhá nad normalizovanými mřížkami, tak metoda podporuje invarianci geometrických a fotometrických transformací. Tyto změny se projeví pouze ve větších prostorových regionech. Kromě toho, jak Dalal a Triggs zpozorovali, hrubé prostorové vzorkování, jemné orientace vzorků a silná místní fotometrická normalizace umožňuje pohyby osob, které budou ignorovány, dokud je zachována zhruba svislá poloha osoby [3]. Deskriptor HOG je tedy vhodný zejména pro detekci osob v obrazech.

Jednotlivé kroky HOG algoritmu:

- Prvním krokem je aplikace globální normalizace, která je určena pro snížení vlivu světelných efektů.
- Druhým krokem je výpočet úhlu a velikosti gradientu v každém pixelu obrazu.
- V třetím kroku jsou gradienty jednotlivých pixelů v lokálním regionu, zvaném mřížka, konvertovány do histogramu.
- Čtvrtým krokem je fáze normalizace. Jednotlivé mřížky jsou normalizovány podle bloku.
- Posledním, patým krokem je sesbírání histogramů ze všech bloků, které obsahuje posuvné okno a jejich následná klasifikace. Zde se rozhoduje zda obraz obsahuje zadaný objekt nebo ne.

4.1 Varianty HoG algoritmu

4.1.1 Rectangular HOG (R-HOG)

Deskriptor R-HOG je založen na překrývajících se obdelníkových nebo čtvercových polích bloků. Blok obsahuje čtvercové nebo obdelníkové pole mřížek. Každý blok se normalizuje samostatně, nezávisle na ostatních blocích. Základními parametry této varinty HOGu jsou: rozměry bloku, zadávané v mřížkách, rozměry mřížek, zadávané v pixelech a počet binů mřížky. R-HOG je podobný SIFT deskriptorům, ale je používán zcela odlišně. R-HOG je počítán na husté síťi, v jednom měřítku a bez zarovnání orientace. Oproti tomu SIFT deskriptory jsou počítány na rozptýlené množině, na rozsahově neměnných bodech

Obrázek 7: Rozdělení obrazu - modrým je znázorněno posuvné okno, zeleným bloky a červeným mřižky [1]

obrazu a přizpůsobené orientaci. R-HOG se používá pro zakódování prostorové informace, zatímco SIFT se používá samostatně. V této práci se pracuje s touto verzí HOG algoritmu.

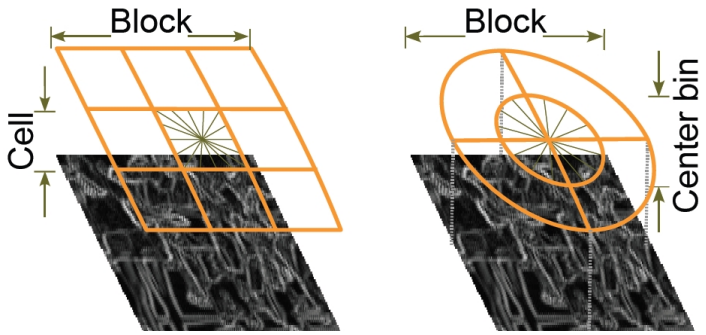
4.1.2 Circular HOG (C-HOG)

V deskriptoru C-HOG jsou základním prvkem bloky kruhového tvaru. Obrázek je pokryt obdelníkovými nebo čtvercovými sítěmi. Každá síť má svůj střed. Tento střed rozdělíme na několik úhlových sektorů a radiálních kanálů. Úhlové sektory společně tvoří kruh, který odpovídá středu bloku. Radiální kanály se počítají na základě logaritmického měřítka. Ve výsledku to znamená, že velikost kanálu roste se vzdáleností od centra. Vlivem této vlastnosti, je v průměru více pixelů ve vnějších mřížkách, než ve vnitřních. Díky vlastnostem této varianty je zbytečné aplikovat na bloky C-HOGu Gaussovu funkci, jelikož nám neposkytuje žádné výhody. Základními paramety C-HOG deskriptoru jsou: počet úhlových sektorů, počet radiálních kanálů, poloměr prostředního kanálu, udávaný v pixelech a rozšiřující faktor pro určení poloměru dalších radiálních kanálů. C-HOG deskriptor je podobný Shape Context deskriptoru, ale liší se v tom, že bloky C-HOGu obsahují buňky s několika orientovanými kanály. Naopak Shape Context počítá pouze jednu hranu.

4.1.3 Centre-Surround HOG

Bloková architektura tohoto deskriptoru je založená na R-HOG deskriptoru. R-HOG aplikuje v každém bodě aktuálního bloku Gaussovo vyvážení a potom vytvoří mřížku, obsahující orientovaný histogram. Pokud bychom neaplikovali Gaussovu váhu, mohl by být výpočet optimalizován tím, že vypočtem orientovaný histogram pro každou buňku. Centrum-Surround HOG umožňuje použít alternativní, středově uzavřený styl normalizace mřížky. Vytvoří se β orientovaných obrazů, kde β odpovídá počtu orientovaných

kanálů. Gradient jednotlivých pixelů je pak začleněn pomocí lineární interpolace do β a uložen do odpovídajícího obrazu. Množina β orientovaných obrazů odpovídá orientovanému histogramu v R-HOG nebo C-HOG.



Obrázek 8: Srovnání R-HOG a C-HOG [2]

4.2 Gama normalizace, normalizace barev

Fáze předzpracování obrazu, používá se k zesvětlení tmavých míst. Mezi používané metody patří gama korekce pomocí druhé odmocniny a log komprese každého barevného kanálu. Dalal a Triggs zjistili, že pro většinu objektů poskytuje lepší výkon gama korekce pomocí druhé odmocniny, oproti nenormalizovaným barevným kanálů. Také zjistili, že druhá odmocnina gama komprese poskytuje lepší výkon při detekci objektů, jako jsou jízdní kola, motocykly, automobily, autobusy a lidé [3]. U detekce zvířat, kde je velká rozmanitost barev a tvarů, se ukazuje být lepší metoda nenormalizovaného RGB obrázku. Zatímco u zvířat s menší rozmanitostí barev a tvarů poskytuje lepší výkon gama korekce pomocí druhé odmocniny.

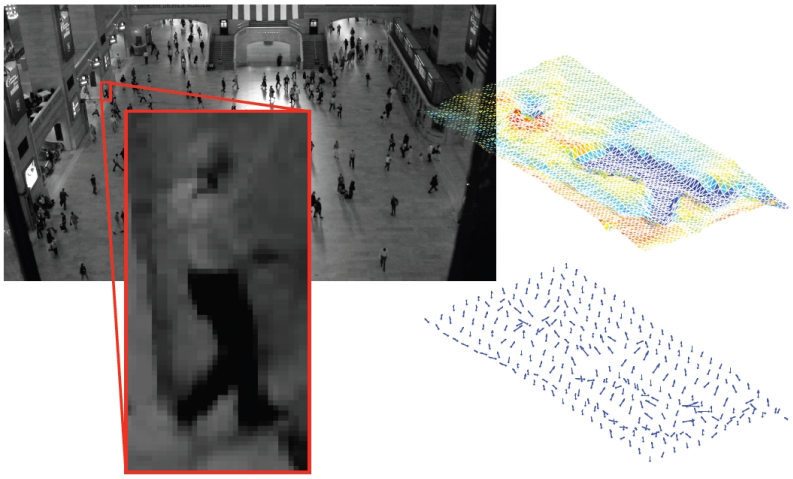
4.3 Výpočet gradientů

V této fázi počítáme velikost a úhel gradientu v každém pixelu zadaného obrazu. U barevných obrázků se vypočítává gradient v každém pixelu, pro každou barvu zvlášť. Pro další výpočty se vezme jen gradient s největší normou a zbylé se zahodí. Nejčastější metoda výpočtu gradientu je použití 1-D bodové diskrétní derivace, buď v jednom směru (horizontální nebo vertikální) nebo v obou směrech. Tato metoda vyžaduje filtr barvy nebo intenzity dat v obraze s tímto jádrem (14):

$$[1, 0, 1] \; a \; [-1, 0, 1]^T \tag{14}$$

Aplikací tohoto jádra získáme x -ovou a y -ovou derivaci hodnoty jasu pixelu. Na základě těchto derivací vypočítáme velikost a úhel gradientu.

Lze použít i jiné metody, které poskytují komplexnější masky, jako například 3×3 Sobel maska nebo diagonální maska, ale Dalal a Triggs zjistili, že tyto masky vykazovali menší výkon oproti 1-D bodové diskrétní derivaci, při detekci osob [3].



Obrázek 9: Ukázka výpočtu gradientů [1]

4.4 Výpočet histogramu

Dalším krokem je výpočet histogramu v každé mřížce zadaného obrazu. Mřížky mohou mít obdelníkový nebo radiální tvar. Délka strany mřížky se udává počtem pixelů. Důležitým parametrem při tvorbě histogramu je počet binů, které jsou rovnoměrně rozloženy do 0 - 180 stupňů v případě neznaménkového gradientu nebo 0 - 360 stupňů v případě znaménkového gradientu. Dalal a Triggs zjistili, že nejlepší výkon pro detekci osob poskytuje kombinace 9-binového histogramu a neznamenkového gradientu [3].

Vstupem pro výpočet histogramu jsou jednotlivé pixely mřížky. Podle úhlu se vypočte bin mřížky, do kterého se gradient zařadí - k velikosti binu se přičtě velikost gradientu vynásobená Gaussovou funkcí, která ma rovnici (15):

$$f(x,y) = Ae^{-(a(x-x_0)^2+2b(x-x_0)(y-y_0)+c(y-y_0)^2)} \tag{15}$$

kde a ,b ,c mají tvar (16) (17) (18):

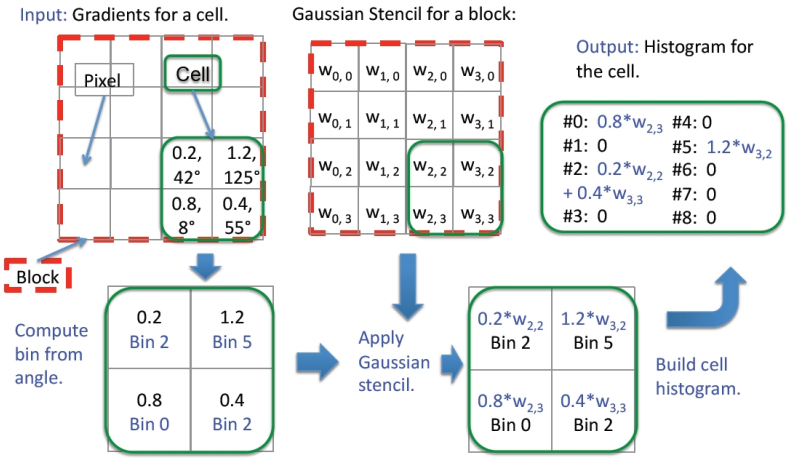
$$a = \frac{cos^2\theta}{2\sigma_x^2} + \frac{sin^2\theta}{2\sigma_y^2} \tag{16}$$

$$b = -\frac{sin2\theta}{4\sigma_x^2} + \frac{sin2\theta}{4\sigma_y^2} \tag{17}$$

$$c = \frac{sin^2\theta}{2\sigma_x^2} + \frac{cos^2\theta}{2\sigma_y^2} \tag{18}$$

a x , y udávají pozici pixelu v mřížce a θ úhel gradientu daného pixelu.

Takto projdem každý pixel v mřížce a zařadíme ho do příslušného binu a tím získáme histogram mřížky. Výpočet je naznačen na obrázku 10.



Obrázek 10: Výpočet histogramu mřížky [1]

4.5 Normalizace

Vlivem změn v osvětlení se může velikost gradientu lišit v mnoha ohledech a proto je potřeba ho místně normalizovat. K tomu je potřeba seskupit mřížky do většího celku, zvaného blok. Každý blok se normalizuje zvlášť. Ve skutečnosti se bloky překrývají a tedy mřížka ovlivňuje více bloků. Dobrá normalizace výrazně zvyšuje výkon algoritmu. Dalal a Triggs při svých pokusech zjistili, že nejlepšího výkonu pro detekci osob se dosahuje při použití mřížky s rozměry 6×6 pixelů, bloku s rozměry 3×3 mřížky a histogramem s 9-ti kanály [3].

Obecný postup normalizace spočívá ve výpočtu normalizační konstanty v každém bloku. Touto konstantou se pak podělí všechny elementy v bloku. Pro výpočet normalizační konstanty se využívají tyto metody:

- L2-norm: $f = \frac{v}{\sqrt{||v||_2^2 + e^2}}$
- L2-hys: vypočtení L2-norm, následováno oříznutím maximálních hodnot v na 0.2 a renormalizací
- L1-norm: $f = \frac{v}{||v||_1 + e}$
- L1-sqrt: $f = \sqrt{\frac{v}{||v||_1 + e}}$

Kde v je nenormalizovaný vektor, obsahující všechny histogramy v daném bloku, $||v_k||$ je k-tá norma vektoru pro k=1,2 a e je malá konstanta, jejíž hodnota se blíží nule.

Dalal a Triggs zjistili, že L2-Hys, L2-norm, a L1-sqrt poskytují zhruba stejný výkon a L1-norm poskytuje o něco menší výkon, avšak všechny čtyři metody poskytují výrazné zlepšení výkonu oproti práci s nenormalizovanými daty [3].

4.6 Klasifikace

Posledním krokem algoritmu je rozhodnutí, zda obrázek obsahuje zadaný objekt, který jsme chtěli detekovat nebo ne. Toho se dosahuje použitím klasifikátoru. V této práci se pro klasifikaci používá Support Vector Machine algoritmus, který zde bude detailněji popsán.

4.6.1 Support Vector Machine

Jedná se o metodu strojového učení používanou pro klasifikaci a regresi. Na základě trénovacích dat, které jsou označeny zda jsou pozitivní nebo negativní (v případě binární klasifikace) se vytváří model, podle kterého se nová data zařazují do příslušné kategorie. Jinak řečeno, algoritmus se snaží najít nadrovinu (lineární funkci), která zadaná data rozděluje do dvou tříd. Ne vždy se dají zadaná data lineárně rozdělit, a proto je někdy potřeba převést původní vstupní prostor do nového, vícedimenzionálního, kde již je možné oddělit data od sebe lineárně - toto se nazývá jádrová transformace. Například dvourozměrný vektor lze přidáním třetího atributu, závislého na předchozích dvou převést do trojrozměrného prostoru. Díky jádrové transformaci jsme schopni původně úlohu, která by vedla k nalezení nelineární hranice převést na lineární úlohu, na kterou aplikujeme algoritmus pro nalezení rozdělující nadroviny. Problémem je, kam nejlíp umístit hranici, aby byla umístěna co nejefektivněji pro začlenění budoucích dat (dat, která nebyla použita pro trénink a u nichž se budem dotazovat, do jaké třídy patří). Optimální rozdělení je definováno jako maximum minima vzdálenosti mezi body z rozdělovaných dat. Toto rozdělění je vidět na obrázku 11, kde zvolíme druhou nadrovinu, zobrazenou plnou čarou, protože nám poskytuje více prostoru na obou stranách než první nadrovina, zobrazená čárkovanou čarou.

Vstupem algoritmu je množina trénovacích dat, ve tvaru (19):

$$D = \{(x_i, c_i)|x_i \in R^p, c_i \in \{-1, 1\}\}_{i=1}^{n} \tag{19}$$

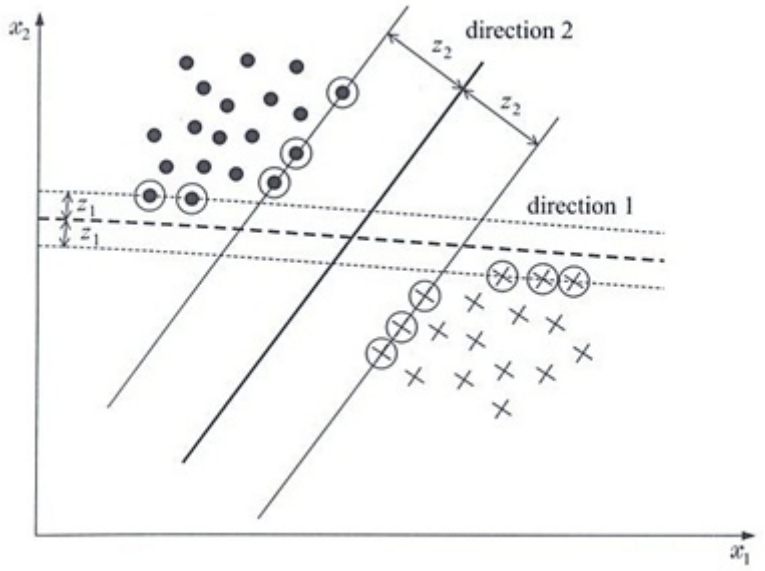
kde c_i je hodnota udavájící, do které třídy patří vektor x_i. c_i nabývá hodnot 1 a -1, přičemž 1 značí pozitivní třídu a -1 negativní. x_i je p-dimenzionální vektor. Hledáme optimální nadrovinu, která rozděluje body z třídy $c_i = -1$ od bodů třídy $c_i = 1$. Tato nadrovina je množina bodů, splňující tuto podmínku (20):

$$w \cdot x - b = 0 \tag{20}$$

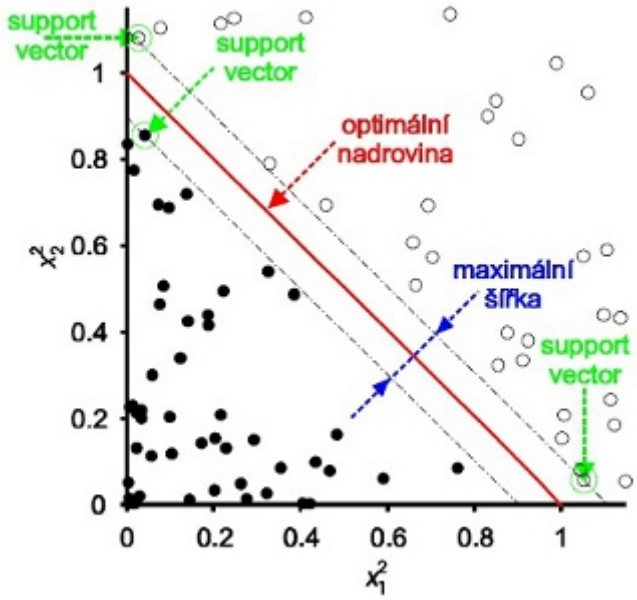
kde $\cdot$ značí skalární součin. w je normálový vektor nadroviny a $\frac{b}{||w||}$ určuje kolmou vzdálenost nadroviny od počátku souřadnicového systému. Potřebujeme najít w a b, tak aby vzdálenost mezi rovnobežnými nadrovinami byla co největší a zároveň, aby oddělovaly data. Nadroviny mohou být popsány následujícími rovnicemi (21) (22):

$$w \cdot x - b = 1 \tag{21}$$

a



Obrázek 11: Hledání optimální nadroviny [37]



Obrázek 12: Příklad optimální nadroviny. Vektory, které jsou nejblíže hranici se nazývají podpůrné vektory [38]

$$w \cdot x - b = -1 \tag{22}$$

Jednotlivé body třídy pak musí splňovat následující podmínku (23) (24):

$$w \cdot x - b \geq 1 \tag{23}$$

pro všechny x_i první třídy

$$w \cdot x - b \leq -1 \tag{24}$$

pro všechny x_i druhé třídy
Tyto dvě rovnice můžem obecně přepsat jako (25):

$$c_i(w \cdot x - b) \geq 1 \tag{25}$$

pro všechny i, splňující podmínku: $1 \geq i \geq n$
Když toto vše dáme dohromady, dostanem optimalizační problém: vektorová geometrie ukazuje, že rozpětí roviny je $\frac{1}{||w||}$, tedy potřebujem minimalizovat $||w||$. Tato minimalizace je zobrazena v rovnici (26):

$$c_i(w \cdot x - b) \geq 1 \tag{26}$$

pro každě i = {1,. . . ,n}
Minimalizace $||w||$ je rovna minimalizaci $\frac{1}{2}||w||^2$. Což umožňuje provádět kvadratické programování. Takže potřebujem minimalizovat $\frac{1}{2}||w||^2$. Dostanem rovnici (27):

$$c_i(w \cdot x - b) \geq 1 \tag{27}$$

Vyjádřením tohoto problému pomocí Lagrangeových multiplikátorů α, kde $\alpha_i \geq 0 \; \forall_i$ dostanem (28) (29) (30):

$$L_P \equiv \frac{1}{2}||w||^2 - \alpha[c_i(x_i \cdot w + b) - 1 \; \forall_i] \tag{28}$$

$$\equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i[c_i(x_i \cdot w + b) - 1] \tag{29}$$

$$\equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i c_i(x_i \cdot w + b) + \sum_{i=1}^{L} \alpha_i \tag{30}$$

Potřebujem najít minimální w ,b a co největší α (přičemž musí platit $\alpha_i \geq 0 \; \forall_i$). Toho dosáhnem rozdělením L_P na základě w a b a nastavením derivace na nulu. Dostanem rovnice (31) (32):

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i c_i x_i \tag{31}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i c_i = 0 \tag{32}$$

Dosazením (31) a (32) do (30) dostanem novou formulaci, závislou na α. Potřebujem maximalizovat (33) (34) (35):

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j c_i c_j x_i \cdot x_j \; kde \; \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i c_i = 0 \tag{33}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j \; kde \; H_{ij} = c_i c_j x_i \cdot x_j \tag{34}$$

$$\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha \; kde \; \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i c_i = 0 \tag{35}$$

Vznikne nová formulace L_D, která je dvojí formou primární L_P
Z hledání minima L_P se tedy stalo hledání maxima L_D. Potřebujem tedy najít (36):

$$max_\alpha [\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha] \; kde \; \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i c_i = 0 \tag{36}$$

Jedná se o kvadratický optimalizační problém. Vyřeší se pomocí kvadratického programování, které nám vrátí α a z rovnice (31) vypočte w. Zbývá tedy dopočítat bt.
Jakákoliv data, splňující rovnici (32), což jsou podpůrné vektory x_S, mají tvar (37):

$$c_s(x_s \cdot w + b) = 1 \tag{37}$$

Dosazením do rovnice (31) dostanem (38):

$$c_s(\sum_{m \in S} \alpha_m c_m x_m \cdot x_s + b) = 1 \tag{38}$$

Kde S značí množinu množinu podpůrných vektorů. S je určeno hledáním indikátoru i kde $\alpha_i > 0$. Vynásobením y_s a následným použitím $y_s^2 = 1$ z rovnic (23) a (24) dostanem (39) (40):

$$y_s^2(\sum_{m \in S} \alpha_m c_m x_m \cdot x_s + b) = y_s \tag{39}$$

$$b = y_s - \sum_{m \in S} \alpha_m c_m x_m \cdot x_s \tag{40}$$

Místo používání libovolného podpúrného vektoru x_s, je lepší vzít průměr všech podpůrných vektorů v množině S (41):

$$b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum \alpha_m y_m x_m \cdot x_s) \tag{41}$$

Touto rovnicí vypočtem zbývající parametr b a tudíž máme oba parametry potřebné k nalezení optimální nadroviny.

5 Implementace

5.1 Knihovna OpenCV

OpenCV je knihovna, určená pro práci s obrazem. Je zaměřena především na zpracování obrazu v reálném čase. Původně byla vyvinuta firmou Intel, ale nyní je tato knihovna volně šířitelná pod licencí BSD. Knihovna je napsána v C/C++ a je multiplatformní, tedy ji lze využít v různých operačních systémech.

5.2 Vlastní datové typy

Jelikož knihovna OpenCV neposkytuje datové typy pro uložení informací o mřížkách, blocích a okně detektoru, tak pro potřebu implementace byly vytvořeny následující vlastní datové typy:

- CELL - struktura, reprezentující mřížku. Řešeno jako pole reálných čísel. Pole má velikost podle počtu binů. Uchovává si informace o velikost jednotlivých binů v mřížce.

```
typedef struct
{
    double hist [ pocetBinu ];
} CELL;
```

Výpis 1: struktura CELL

- BLOCK - struktura, reprezentující blok. Vnitřně se jedná o dvourozměrné pole datového typu CELL.

```
typedef struct
{
    CELL** c;
} BLOCK;
```

Výpis 2: struktura BLOCK

- DETECT_WINDOW - struktura, reprezentující okno detektoru. Vnitřně si uchováná dvourozměrné pole datového typu BLOCK. Tato datová struktura slouží pro uložení normalizovaných hodnot mřížek, pro jejich následnou klasifikaci.

```
typedef struct
{
    BLOCK** b;
} DETECT_WINDOW;
```

Výpis 3: struktura DETECT_WINDOW

5.3 Postup implementace

5.3.1 Načtení obrázku

Prvním krokem je načtení obrázku pomocí funkce cvLoadImage(const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR). Funkce vrací v případě úspešného načtení ukazatel na obrázek, v případě neúspěchu NULL. Tento obrázek je následně pomocí funkce cvCvtColor(const CvArr* src, CvArr* dst, int code) převeden do černobílého obrázku, s kterým se dále pracuje.

5.3.2 Gama normalizace, normalizace barev

Pro gama korekci je zde použita vlastní funkce - GamaKorekce(IplImage *img, const float gama). Která projde všechny pixely zadaného obrazu a upraví je dle zadané konstanty. Funkce vrací normalizovaný obrázek. Podrobnější teoretický popis této fáze je popsán v kapitole 4.2.

5.3.3 Výpočet gradientů

K výpočtu úhlu a velikosti gradientů nejprve potřebujem vypočítat derivace. Ty vypočtem pomocí funkce cvSobel(const CvArr* src, CvArr* dst, int xorder, int yorder, int apertureSize=3), kterou nám poskytuje knihovna OpenCV. Použijeme ji na výpočet první x-ové a poté první y-ové derivace. Poté použijeme další funkci z knihovny OpenCV - cvCartToPolar(const CvArr* x, const CvArr* y, CvArr* magnitude, CvArr* angle=NULL, int angleInDegrees=0), která nám vypočte úhel a velikost jednotlivých gradientů a tyto údaje uloží do zadaných polí. Teoretickým popisem této fáze se zabývá kapitola 4.3.

5.3.4 Výpočet histogramu

Pro výpočet histogramu byla vytvořena vlastní funkce - VypoctiHistogram(CELL** poleMrizek, int vyskaObrazku, int sirkaObrazku, CvMat* poleUhlu, CvMat* poleVelikosti, int sirkaMrizky, int pocetBinu, int maximalniUhel). Její návratový typ je void. Funkce projde celý obrázek, pixel po pixelu a do pole, zadaného parametrem funkce uloží hodnoty binů jednotlivých mřížek v zadaném obrázku.

Pro vykreslení histogramu je potřeba spočítat bin s maximální hodnotou v jednotlivých mřížkách. K tomu byla vytvořena vlastní funkce - VypoctiMaximaMrizek(CELL** poleMrizek, int** poleMaxHodnot, int pocetMrizekX, int pocetMrizekY, int pocetBinu). Jako jeden z parametrů bere pole mřížek. Funkce projde všechny mřížky ze zadaného pole a do pole, zadaného parametrem funkce uloží index binu s maximální hodnotou.

Samotné vykreslení probíhá pomocí funkce VykresliHistogram(int pocetMrizekX, int pocetMrizekY, int sirkaMrizky, int pocetBinu, int** histMax). Tato funkce vykreslí jednotlivé mřížky a maximální hodnotu binu v jednotlivých mřížkách. Funkce vrací obrázek, představující histogram zadaného obrazu. Podrobnější teoretický popis této fáze je popsán v kapitole 4.4.

5.3.5 Normalizace

Pro aplikaci normalizace byly vytvořeny dvě vlastní funkce. První - NormalizacniKonstanta(BLOCK blok, int sirkaBloku, int pocetBinu), slouží pro výpočet normalizační konstanty z bloku, zadaného jako parametr funkce. Funkce vrací hodnotu normalizační konstanty. Pro výpočet je použitá metoda L2-norm. Druhá - NormalizujBlok(BLOCK blok, int sirkaBloku, int pocetBinu, double normKonstanta) je funkce provádějící samotnou normalizaci. Tato funkce projde všechny mřížky v daném bloku, znormalizuje je normalizační konstantou, zadanou jako parametr funkce a normalizované mřížky uloží do proměnné, zadané parametrem funkce, představující blok. Podrobnější teoretický rozbor normalizace je popsán v kapitole 4.5.

5.3.6 Klasifikace

Pro klasifikace je zde, jak již bylo psáno výše, použit algoritmus Support Vector Machine, popsán v kapitole 4.6.1. Je zde použita implementace z knihovny OpenCV. Po nastavení patřičných parametrů klasifikátoru, je potřeba zadat trénovací data. Tedy data, u kterých víme zda jsou pozitivní nebo negativní a na kterých budeme učit náš klasifikátor rozdělovat zadaná data. Pro klasifikaci potřebujem zadat pole samotných dat a pole určující rozdělení dat do tříd (pozitivní nebo negativní).

Pro zadání trénovací dat v aplikaci máme dvě možnosti. První je, zadat adresář obsahující pozitivní obrázky a poté zadat adresář s negativními daty. Jednotlivé obrázky se projdou, vytvoří se jejich histogram a ten se uloží do pole, představujícího trénovací data. Zároveň se do pole, reprezentující třídy uloží třída jednotlivých obrázku. Nakonec je potřeba zadat jméno souboru, do kterého se mají tréninková data uložit. Soubor má přesně zadanou zadanou strukturu. Na prvním řádku je zapsán počet trénovacích obrázku. Na druhém a třetím řádku jsou zapsány výška a šířka okna detektoru. Rozměry okna jsou udavány v mřížkách. Pak jsou postupně zapsány hodnoty histogramu jednotlivých obrázků. Po zápisu histogramu obrázku vždy následuje zapsání klasifikace obrázku. Pro pozitivní se zapisuje 1, pro negativní -1. Každá hodnota je uložena na samostatném řádku.

Druhou možností je načtení dat ze souboru. Soubor musí mít výše zadanou strukturu. V opačném případě nedojde k načtení dat nebo ke špatnému načtení. Pro vlastní načtení ze souboru slouží vlastní funkce - void NactiTrenovaciData(char* nazevSouboru, float** tData, int** tTrida, int* pocetTObrazku, int* vyskaOkna, int* sirkaOkna, int pocetBinu). Ze souboru, zadaným jako parametr funkce, se do proměnných, zadaných jako parametr funkce, načtou samotná trénovací data, zařazení těchto dat do tříd, počet trénovacích obrázků a rozměry okna detektoru. Při neúspešném načtení funkce vypíše chybu.

Pokud máme zadané nebo načtené tréninková data, můžem pomocí OpenCV funkce train(const CvMat* _train_data, const CvMat* _responses, const CvMat* _var_idx=0, const CvMat* _sample_idx=0, CvSVMParams _params=CvSVMParams()) naučit rozpoznávat obrázky. Funkce požaduje pole trénovacích dat, pole zařazení těchto dat do tříd a parametry klasifikace.

Nyní už zbývá jen samotná klasifikace zadaného obrázku. Pro zadaný obrázek projdem všechny detekující okna v obrázku. Histogram jednotlivých oken si uložíme do matice a pomocí OpenCV funkce predict(const CvMat* _sample, bool returnDFVal=false), která jako parametr požaduje matici, představující histogram, nám klasifikátor rozhodne o zařazení zadaného okna. Funkce vrací reálné číslo, představující třídu, do které okno patří. Podle výsledků klasifikace jednotlivých oken se pak rozhodne, zda obrázek obsahuje zadaný objekt, který jsme v obrázku hledali nebo ne. Pokud obrázek obsahuje alespoň jedno pozitivní okno, tak se na zadaném obrázku nachází požadovaný objekt.

6 Testování

Testování aplikace probíhalo systematicky, od detekce nejjednodušších objektů v mřížkách až po samotnou detekci osob v celém obraze.

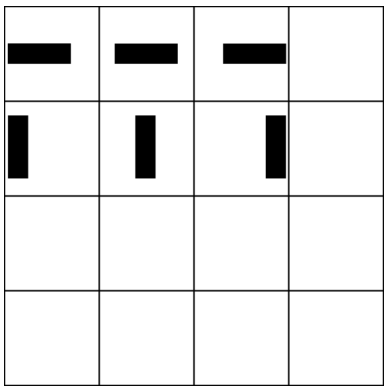
Nejpve byl jako testovací objekt zvolen váleček. Mřížky s vodorovnými válečky představovaly pozitivní trénovací objekty a mřížky se svislými válečky představovaly negativní trénovací objekty. Trénovací objekty byly posouvány po ose x a y, ale neměnily se jejich rozměry. Do trénovacích objektů byly nejprve zahrnuty jen válečky, které byly celé v mřížce. Příklad těchto válečků je na obrázku 13. Okno detektoru mělo velikost 1×1 mřížka. Samotné testování pak probíhalo procházením jednotlivých mřížek a porovnání jejich histogramu s tréninkovými daty. Průměrná detekce úspěšnost této sekce byla 95%. Výsledky ukazuje tabulka 1.

Po otestování funkčnosti detekce celých válečků pak byly přidány i válečky, které byly tak posunuté, že nebyly celé v mřížce. Trénovací data jsou zobrazeny na obrázku 14. Výsledky jsou zobrazeny v tabulce 2.

V další fázi bylo zvětšeno okno detektoru na rozměry 2×1 mřížky. Jako trénovací data posloužily válečky, uvedené výše. Okno s vodorovnými válečky jako pozitivní a okno se svislými válečky jako negativní. Prvním krokem bylo testování detekce válečků, které byly celé v jedné mřížce, tedy součástí trénovacích dat, ani samotné detekce nebyly válečky, které byly přes dvě mřížky. Výsledky této fáze byly výborné - vše detekoval správně. Poté byly do trénovacích dat přidány i válečky, které byly přes dvě mřížky. Při detekci válečků, které byly přes dvě mřižky, nastaly první problémy. Úspešnost detekce těchto objektů se pohybovala okolo 50%. Klasifikátor nebyl schopen správně detekovat válečky, které byly přes dvě mřížky a byly trošku posunuty po ose x vůči tréninkovým objektům. Bylo tedy přidáno do tréninkových dat více válečků, které byly přes dvě mřížky. Výsledky následné detekce již byly daleko příznivější. Úspěšnost detekce se pohybovala okolo 90%.

Po úspešném otestování detekce válečků se přešlo na otestování složitějších objektů. Těmito objekty byli panáčci sestavení z válečků. Pozitivní data byly obrázky, na kterých byli panáčci, různě posunutí po osách x a y a negativní data byly obrázky, na kterých nebyli panáčci - zde byly použity válečky. Na obrázku 15 je zobrazen panáček, kterého jsem se snažil detekovat. Detekující okno mělo rozměry 2×3 mřížky. Zde se opět objevil problém z předešlého testování - klasifikátor měl problémy s objekty, které byly přes více mřížek. Přidáním dalších pozitivních trénovacích dat se tento problém vyřešil a výkonnost stoupla z původních 50% na 90%. Z toho tedy plyne, že při špatné výkonnosti klasifikátoru je potřeba přidat více trénovacích dat. Výsledky této části jsou zobrazeny v tabulce 3.

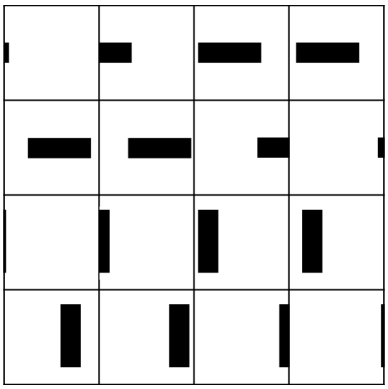
Při testování bylo zjištěno, že pro správnou a objektivní klasifikaci, je potřeba mít stejný počet pozitivních a negativních trénovacích dat. V případě, že pozitivních nebo negativních trénovacích dat bude více, je možné, že detekce bude fungovat správně, ale výsledek nebude objektivní. Pro správnou funkčnost aplikace je potřeba, aby výška i šířka detekujícího okna (v mřížkách), byly beze zbytku dělitelné šířkou bloku (zadána jako počet mřížek). Také je potřeba při nastavování konfigurace dát pozor, aby výška i šířka obrazu (v pixelech) byly beze zbytku dělitelné šiřkou mřížky, zadanou v pixelech.



Obrázek 13: Testování detekce celých válečků

Pozitivní trénovací obrázky	3
Negativní trénovací obrázky	3
Průměrná úspěšnost klasifikace	95%

Tabulka 1: Testování detekce celých válečků



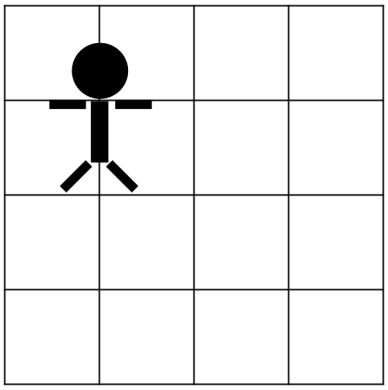
Obrázek 14: Trénovací data, zobrazující celé i částečné válečky

Pozitivní trénovací obrázky	8
Negativní trénovací obrázky	8
Průměrná úspěšnost klasifikace	92%

Tabulka 2: Testování detekce celých i částečných válečků

Pozitivní trénovací obrázky	10
Negativní trénovací obrázky	10
Průměrná úspěšnost klasifikace	90%

Tabulka 3: Testování detekce panáčků.



Obrázek 15: Ukázka panáčka, kterého jsem se snažil detekovat.

Obrázek 16: Příklad pozitivních trénovacích obrázků z MIT Pedestrian databáze

Po důkladném otestování klasifikátoru přichází na řadu samotná detekce osob. Jako pozitivní trénovací obrázky byly použity obrázky z databáze MIT Pedestrian databáze, příklad je na obrázku 16. Jako negativní trénovací obrázky lze použít jakýkoliv obrázek, na kterém není osoba. Nejprve byly jako negativní obrázky použity válečky. Výsledky detekce shrnují tabulky 4 a 5. Poté byly jako negativní obrázky, použity reálné fotografie, na kterých nebyli lidé, z INRIA databáze, zobrazeny na obrázku 17. Výsledky jsou zobrazeny v tabulkách 6 a 7.

V této závěrečné fázi testování nastaly drobné problémy, že klasifikátor si nedokázal poradit s detekcí zadaných objektů a klasifikoval je špatně. Pro zlepšení výkonu detekce a odstranění tohoto drobného problému by bylo potřeba zadat více trénovacích dat a také najít optimální nastavení šířky mřížky a šířky bloku.

Pozitivní trénovací obrázky	20
Negativní trénovací obrázky	20
Průměrná úspěšnost klasifikace	64%

Tabulka 4: Testování detekce osob, oproti válečkům. Konfigurace: šířka mřížky: 16, šířka bloku: 2.

Obrázek 17: Příklad negativních trénovacích obrázků z INRIA databáze

Pozitivní trénovací obrázky	60
Negativní trénovací obrázky	60
Průměrná úspěšnost klasifikace	50%

Tabulka 5: Testování detekce osob, oproti válečkům. Konfigurace: šířka mřížky: 4, šířka bloku: 4.

Pozitivní trénovací obrázky	100
Negativní trénovací obrázky	100
Průměrná úspěšnost klasifikace	50%

Tabulka 6: Testování detekce osob, oproti negativním obrázkům z INRIA databáze. Konfigurace: šířka mřížky: 4, šířka bloku: 2.

Pozitivní trénovací obrázky	100
Negativní trénovací obrázky	100
Průměrná úspěšnost klasifikace	50%

Tabulka 7: Testování detekce osob, oproti negativním obrázkům z INRIA databáze. Konfigurace: šířka mřížky: 4, šířka bloku: 4.

7 Závěr

Cílem této práce bakalářské práce bylo popsat co je to deskriptor, k čemu slouží a naimplementovat vybraný deskriptor pro detekci objektů v obraze. Vybral jsem si deskriptor Histogram of Oriented Gradients, jehož jednotlivé kroky jsou v této práci podrobně popsány, včetně klasifikace pomocí Support Vector Machine. Snažil jsem se implementaci zaměřit co nejvíce na použitelnost - načítání trénovacích dat ze souboru, možnost změnit si rozměry mřížky, bloku a počet binů. Vlastní implementace tohoto deskriptoru není úplně optimální, jelikož se pracuje se s obrázkem, převedeným do černobílé stupnice. Přesto aplikace dosahala dobrých výsledků při testování detekce objektů. Případné další vylepšení této implementace by spočívalo v práci s barevným obrázkem namisto černobílého a použití post-processingu, který zabrání multidetekci objektu v obraze. Pro zrychlení výpočtu by bylo možno převést výpočet na GPU. Tedy použít pro implementaci technologii CUDA.

8 Reference

[1] Gerald Dalley, Krista Ehinger, Keza Covacs, Jim Sukha *Pedestrian Detection in High Definition Video Frames*, Massachusetts Institute of Technology

[2] Navneet Dalal *Finding People in Images and Videos*, 2006

[3] Navneet Dalal, Bill Triggs *Histograms of Oriented Gradients for Human Detection*

[4] Nathan Sprague, Jiebo Luo *Clothed People Detection in Still Images*

[5] F. Suard, A. Rakotomamonjy, A. Bensrhair, A. Broggi *Pedestrian Detection using Infrared images and Histograms of Oriented Gradients*, 2006

[6] Bernt Schiele, Mykhaylo Andriluka, Nikodem Majer, Stefan Roth, Christian Wojek *Visual People Detection – Different Models, Comparison and Discussion*, 2009

[7] Tristan Fletcher *Support Vector Machines Explained*, 2009

[8] Serge Belongie, Jitendra Malik, Jan Puzicha *Shape Matching and Object Recognition Using Shape Contexts*, 2002

[9] Greg Mori, Jitendra Malik *Estimating Human Body Configurations using Shape Context Matching*

[10] Luo Juan, Oubong Gwun *A Comparison of SIFT, PCA-SIFT and SURF*

[11] David G. Lowe *Distinctive Image Features from Scale-Invariant Keypoints*, 2004

[12] cs.wikipedia.org - článek Barevný histogram, 6.4.2010

[13] cs.wikipedia.org - článek Gradient, 6.4.2010

[14] cs.wikipedia.org - článek Parciální derivace, 6.4.2010

[15] cs.wikipedia.org - článek Skalární pole, 6.4.2010

[16] cs.wikipedia.org - článek Skalár, 6.4.2010

[17] cs.wikipedia.org - článek Support vector machines, 10.4.2010

[18] cs.wikipedia.org - článek OpenCV, 11.4.2010

[19] en.wikipedia.org - článek Visual descriptors, 6.4.2010

[20] en.wikipedia.org - článek Feature detection (computer vision), 20.3.2010

[21] en.wikipedia.org - článek Blob detection, 20.3.2010

[22] en.wikipedia.org - článek Difference of Gaussians, 21.3.2010

[23] en.wikipedia.org - článek Scale-invariant featuretransform, 11.4.2010

[24] en.wikipedia.org - článek Image gradient, 6.4.2010

[25] en.wikipedia.org - článek Histogram of oriented gradients, 20.3.2010

[26] en.wikipedia.org - článek Gaussian function, 20.3.2010

[27] en.wikipedia.org - článek Support vector machine, 10.4.2010

[28] en.wikipedia.org - článek OpenCV, 11.4.2010

[29] en.wikipedia.org - článek Shape context, 11.4.2010

[30] manual.gimp.org/en/filters.html - filtr Laplace, 11.4.2010

[31] manual.gimp.org/en/filters.html - filtr Difference of Gaussians, 11.4.2010

[32] Difference of Gaussian (DoG) - fourier.eng.hmc.edu/e161/lectures/gradient/node11.html, 21.3.2010

[33] Laplacian/Laplacian of Gaussian - homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm, 21.3.2010

[34] Histogram a jeho praktické použití - fotoroman.cz/techniques2/exposure_histo.htm, 6.4.2010

[35] Vše o světle – 13. Histogram - www.fotografovani.cz/art/fozak_df/rom_1_13_histogram.html, 6.4.2010

[36] Histogram - www.fotoroman.cz/glossary2/3_histogram.htm, 6.4.2010

[37] Lineární klasifikátory - cmp.felk.cvut.cz/cmp/courses/recognition/resources/course_stanclova_pattrec/06%20-%20Linearni_klasifikator.ppt, 10.4.2010

[38] Support vector machines (SVM) - is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf?fakulta=1433;obdobi=3523;kod=PA034, 10.4.2010